```
/*
 * Programm zur Animation einer aus Epoxydharz gegossenen Kerzenflamme
 * Grundlage war das Projekt "Stimmungslaterne" von Tobias Kuch im Blog bei AZ-Delivery
 *  Änderungen:   - Keine Fernbedienung
 *               - Feuersequenzen per 3poligem DIP-Switch einstellbar
 *                - Lauffähig auf ATTiny85-20
 *
 * Projektbeschreibung:
 * Meine Kerzenflamme ist etwa 17cm hoch, 10cm breit und etwa 9mm dick. Mit Gießharz gegossen in einer
 * eigens dafür hergestellten Silikonform. Für bessere Refektionen habe ich Goldglitzer-Partikel mit
 * eingemischt. Die Epoxydflamme steckt in einem passenden Ausschnitt in einem Brett und
 * wird von unten mit 7 Neopixel (aus Sreifen mit 144LEDs/Meter) angefeuert.
 *
 * Benötigte Librarys   'Adafruit Neopixel' >= 1.3.5
 *
 * Für ATTiny-Kompilat zusätzlichen Boardverwalter http://drazzy.com/package_drazzy.com_index.json
 *                     Board: 'AttinyCore' by Spence Konde V>= 1.4.1
 *    Für ATTiny 85-20  Vorbereitung (Nur einmal pro Prozessor) Board:  1. Attiny25/45/85(no Bootloader)
 *                                                                      2. Serialport einstellen
 *                                                                     3. Programmer: Arduino as ISP (AttinyCore)
 *                                                                      4. Bootloader brennen
 *
 *                      Programm übertragen                             1. ATTinyCore einstellen (Attiny
25/45/85(no Bootloader)
 *                                                                       2. Chip ATTiny85
 *                                                                       3. 8MHz intern
 *                                                                      4. Programmer: Arduino as ISP (AttinyCore)
 *                                                                      5. Programm 'hochladen mit Programmer'
 */

#define ATTINY               // aktivieren, wenn für ATTiny kompiliert werden soll

#include<Adafruit_NeoPixel.h>

#ifdef ATTINY                  // Definitionen für ATTiny 85-20

  #define LEDPIN  4           // Which pin on the ATTiny is connected to the NeoPixels? (Attiny-Chip Pin 3)
  #define RANDOM_INIT_PIN 3    // Analog-Input for init Random Generator (ATTiny-Chip Pin 2)
  #define DIL_SWITCH0 2        // Bit 0 DILSwitch to select FireSequence (ATTiny-Chip Pin 5)
  #define DIL_SWITCH1 1        // Bit 1 DILSwitch to select FireSequence (ATTiny-Chip Pin 6)
  #define DIL_SWITCH2 0        // Bit 2 DILSwitch to select FireSequence (ATTiny-Chip Pin 7)

#else                          // Definitionen für Arduino

  #define LEDPIN  13           // Which pin on the Arduino is connected to the NeoPixels?
  #define RANDOM_INIT_PIN 0    // Analog-Input for init Random Generator
  #define DIL_SWITCH0 7        // Bit 0 DILSwitch to select FireSequence
  #define DIL_SWITCH1 6        // Bit 1 DILSwitch to select FireSequence
  #define DIL_SWITCH2 5        // Bit 2 DILSwitch to select FireSequence
#endif


#define NUMPIXELS 7  // How many NeoPixels are attached to the Arduino? // Popular NeoPixel ring size
Adafruit_NeoPixel pixels(NUMPIXELS, LEDPIN, NEO_GRB + NEO_KHZ800);



long FirelastTime = 0;
int interval;
byte FireSequence=0;

void setup()
{
 //Serial.begin(115200);
  //while (!Serial);   //wait until Serial is established - required on some Platforms

 pinMode(DIL_SWITCH0, INPUT_PULLUP);
 pinMode(DIL_SWITCH1, INPUT_PULLUP);
 pinMode(DIL_SWITCH2, INPUT_PULLUP);
```

```
 pixels.begin(); // INITIALIZE NeoPixels
 pixels.show(); // Initialize all pixels to 'off'
 interval = 300;
randomSeed(analogRead(RANDOM_INIT_PIN));

 //Serial.println(FireSequence);
}

void SimulateFire (byte FireSq)
{
 byte LightValue[NUMPIXELS * 3];

 if (millis() - FirelastTime >= interval)
 {
  FirelastTime = millis();
  interval = 200;

  if (FireSq == 0)                        // Orange ohne flackern
  {
    for (int i = 0; i < NUMPIXELS; i++)
    { // For each pixel...
      pixels.setPixelColor(i, 250, 50, 0);
    }
   pixels.show();   // Send the updated pixel colors to the hardware.
  }

  if (FireSq == 1)                        // Orange leichtes flackern
  {
   for (int i = 0; i < NUMPIXELS; i++)
   { // For each pixel...
    LightValue[i * 3] = random(200, 255); // 250
    LightValue[i * 3 + 1] = random(30, 70); // 50
    LightValue[i * 3 + 2] = 0;
   }
   for (int i = 0; i < NUMPIXELS; i++)
   { // For each pixel...
    pixels.setPixelColor(i, LightValue[i * 3], LightValue[i * 3 + 1], LightValue[i * 3 + 2]);
   }
   pixels.show();   // Send the updated pixel colors to the hardware.
  }

  if (FireSq == 2)                        // Orange starkes flackern kleine Abstände
  {
   interval = random(50, 100);
   for (int i = 0; i < NUMPIXELS; i++)
   { // For each pixel...
    LightValue[i * 3] = random(240, 255); // 250
    LightValue[i * 3 + 1] = random(30, 60); // 50
    LightValue[i * 3 + 2] = 0;
   }
   // Switch some lights out
   byte LightsOff  = random(0, 6);
   for (int i = 0; i < LightsOff; i++)
   {
    byte Selected = random(NUMPIXELS);
    LightValue[Selected * 3] = 0;
    LightValue[Selected * 3 + 1] = 0;
    LightValue[Selected * 3 + 2] = 0;
   }
   for (int i = 0; i < NUMPIXELS; i++)
   { // For each pixel...
    pixels.setPixelColor(i, LightValue[i * 3], LightValue[i * 3 + 1], LightValue[i * 3 + 2]);
   }
   pixels.show();   // Send the updated pixel colors to the hardware.
  }

  if (FireSq == 3)                        // Orange starkes flackern große Abstände
  {
```

```
interval = random(80);
for (int i = 0; i < NUMPIXELS; i++)
{ // For each pixel...
  LightValue[i * 3] = 250; //random(240,255); // 250
  LightValue[i * 3 + 1] =  50; //random(30,60); // 50
   LightValue[i * 3 + 2] = 0;
}
// Switch some lights out if Chance Hit

byte ChanceforLightsOff  = random(0, 40);
if (ChanceforLightsOff > 35)
{
  byte LightsOff  = random(5);
  for (int i = 0; i < LightsOff; i++)
   {
    byte Selected = random(NUMPIXELS);
    LightValue[Selected * 3] = 0;
    LightValue[Selected * 3 + 1] = 0;
    LightValue[Selected * 3 + 2] = 0;
   }
}
for (int i = 0; i < NUMPIXELS; i++)
{ // For each pixel...
   pixels.setPixelColor(i, LightValue[i * 3], LightValue[i * 3 + 1], LightValue[i * 3 + 2]);
  }
 pixels.show();   // Send the updated pixel colors to the hardware.
}

if (FireSq == 4)                                  // Kaminfeuer
{
  interval = random(150, 200);
  for (int i = 0; i < NUMPIXELS; i++)
  { // For each pixel...
    LightValue[i * 3] = random(240, 255); // 250
    LightValue[i * 3 + 1] = random(30, 60); // 50
    LightValue[i * 3 + 2] = 0;
  }
  // Switch some lights darker
  byte LightsOff  = random(0, 4);
  for (int i = 0; i < LightsOff; i++)
  {
    byte Selected = random(NUMPIXELS);
    LightValue[Selected * 3] = random(50, 60);
    LightValue[Selected * 3 + 1] = random(5, 10);
    LightValue[Selected * 3 + 2] = 0;
  }
  for (int i = 0; i < NUMPIXELS; i++)
  { // For each pixel...
    pixels.setPixelColor(i, LightValue[i * 3], LightValue[i * 3 + 1], LightValue[i * 3 + 2]);
  }
 pixels.show();   // Send the updated pixel colors to the hardware.
}

if (FireSq == 5)                          // heftiges kurzes flackern in gelb
{
  interval = random(50, 100);
  for (int i = 0; i < NUMPIXELS; i++)
  { // For each pixel...
    LightValue[i * 3] = random(240, 255); // 250
    LightValue[i * 3 + 1] = random(100, 160); // 50
    LightValue[i * 3 + 2] = 0;
  }
  // Switch some lights out
  byte LightsOff  = random(0, 6);
  for (int i = 0; i < LightsOff; i++)
  {
    byte Selected = random(NUMPIXELS);
    LightValue[Selected * 3] = 0;
    LightValue[Selected * 3 + 1] = 0;
```

```
    LightValue[Selected * 3 + 2] = 0;
   }
  for (int i = 0; i < NUMPIXELS; i++)
  { // For each pixel...
   pixels.setPixelColor(i, LightValue[i * 3], LightValue[i * 3 + 1], LightValue[i * 3 + 2]);
   }
  pixels.show();   // Send the updated pixel colors to the hardware.
}

if (FireSq == 6)                        // leichtes flackern gelb
{
  for (int i = 0; i < NUMPIXELS; i++)
  { // For each pixel...
    LightValue[i * 3] = random(200, 255); // 250
    LightValue[i * 3 + 1] = random(100,160); // 50
     LightValue[i * 3 + 2] = 0;
   }
  for (int i = 0; i < NUMPIXELS; i++)
  { // For each pixel...
   pixels.setPixelColor(i, LightValue[i * 3], LightValue[i * 3 + 1], LightValue[i * 3 + 2]);
   }
  pixels.show();   // Send the updated pixel colors to the hardware.
}

   if (FireSq == 7)                        // leichtes flackern grün
   {
     for (int i = 0; i < NUMPIXELS; i++)
     { // For each pixel...
       LightValue[i * 3] = random(0,50);
       LightValue[i * 3 + 1] = random(130,200);
        LightValue[i * 3 + 2] = 0;
      }
     for (int i = 0; i < NUMPIXELS; i++)
     { // For each pixel...
      pixels.setPixelColor(i, LightValue[i * 3], LightValue[i * 3 + 1], LightValue[i * 3 + 2]);
      }
     pixels.show();   // Send the updated pixel colors to the hardware.
   }

  }
}

void loop()
{
 ReadDIPSwitch();
 SimulateFire(FireSequence);
}

void ReadDIPSwitch()
{
 FireSequence=0;
 if (digitalRead(DIL_SWITCH0)==0) {FireSequence=FireSequence+1;}  // Bit0 addieren, falls gesetzt
 if (digitalRead(DIL_SWITCH1)==0) {FireSequence=FireSequence+2;}  // Bit1 addieren, falls gesetzt
 if (digitalRead(DIL_SWITCH2)==0) {FireSequence=FireSequence+4;}  // Bit2 addieren, falls gesetzt
}
```